What is claimed is:

1. A method of ordering program code in a computer memory, the method comprising:

selecting an ordering from among a plurality of orderings for a plurality of program code segments using a heuristic algorithm; and

ordering the plurality of program code segments in a memory of a computer using the selected ordering.

2. The method of claim 1, wherein the heuristic algorithm is configured to minimize cache misses in the computer.

3. The method of claim 1, wherein the heuristic algorithm comprises a simulated annealing algorithm.

4. The method of claim 3, wherein selecting the ordering using the heuristic algorithm includes testing a subset of the plurality of orderings.

5. The method of claim 4, wherein testing the subset of the plurality of orderings includes, for each ordering in the subset, calculating a cost for such ordering based upon cache miss rates for such ordering.

6. The method of claim 5, wherein calculating the cost for each ordering comprises calculating a plurality of hits/reference values, misses/address values, and misses/entry values.

7. The method of claim 5, wherein testing the subset of orderings includes randomly selecting a different ordering after testing an ordering from the subset of orderings.

8. The method of claim 7, wherein randomly selecting the different ordering comprises swapping two program code segments in a previous ordering.

9. The method of claim 8, wherein the program code segments each comprise a module, and wherein randomly selecting the different ordering further comprises constraining selection of the two program code segments to modules in the same replaceable unit destination.

10. The method of claim 3, wherein selecting an ordering from among the plurality of orderings comprises testing a subset of orderings at each of a plurality of temperature values.

11. The method of claim 10, wherein selecting an ordering from among the plurality of orderings further comprises testing a subset of orderings at each temperature value.

12. The method of claim 11, wherein selecting an ordering from among the plurality of orderings further comprises accepting a change to an ordering if a calculated cost for such ordering is lower than that of a working ordering.

13. The method of claim 11, wherein selecting an ordering from among the plurality of orderings further comprises randomly accepting a change to an ordering even if the calculated cost for such ordering is not lower than that of the working ordering.

14. The method of claim 11, wherein selecting an ordering from among the plurality of orderings further comprises prematurely halting the testing of orderings based upon a halt criterion.

15. The method of claim 1, wherein the program code segments each comprise a module from an operating system kernel.

16. The method of claim 15, wherein each module comprises a high use module, and wherein selecting the ordering from among a plurality of orderings comprises generating a high use module list.

17. An apparatus, comprising:

a processor; and

program code configured to be executed by the processor to optimize execution of program code in a computer of the type including a multi-level memory architecture by using a heuristic algorithm to select an ordering from among a plurality of orderings for a plurality of program code segments in the program code.

18. The apparatus of claim 17, wherein the heuristic algorithm is configured to minimize cache misses in the computer.

19. The apparatus of claim 17, wherein the heuristic algorithm comprises a simulated annealing algorithm.

20. The apparatus of claim 19, wherein the program code is configured to select the ordering using the heuristic algorithm by testing a subset of the plurality of orderings, and wherein the program code is configured to test the subset of the plurality of orderings by, for each ordering in the subset, calculating a cost for such ordering based upon cache miss rates for such ordering.

21. The apparatus of claim 20, wherein the program code is configured to test the subset of orderings by randomly selecting a different ordering after testing an ordering from the subset of orderings.

22. The apparatus of claim 21, wherein the program code is configured to randomly select the different ordering by swapping two program code segments in a previous ordering.

23. The apparatus of claim 22, wherein the program code segments each comprise a module, and wherein the program code is configured to randomly select the different ordering by constraining selection of the two program code segments to modules in the same replaceable unit destination.

24. The apparatus of claim 19, wherein the program code is configured to select an ordering from among the plurality of orderings by testing a subset of orderings at each of a plurality of temperature values, and testing a subset of orderings at each temperature value.

25. The apparatus of claim 24, wherein the program code is configured to select an ordering from among the plurality of orderings by accepting a change to an ordering if a calculated cost for such ordering is lower than that of a working ordering.

26. The apparatus of claim 25, wherein the program code is configured to select an ordering from among the plurality of orderings by randomly accepting a change to an ordering even if the calculated cost for such ordering is not lower than that of the working ordering.

27. The apparatus of claim 17, wherein the program code is configured to select an ordering from among the plurality of orderings by prematurely halting the testing of orderings based upon a halt criterion.

28. The apparatus of claim 17, wherein the program code segments each comprise a module from an operating system kernel.

29. The apparatus of claim 28, wherein each module comprises a high use module, and wherein the program code is configured to select the ordering from among a plurality of orderings by generating a high use module list.

30. A program product, comprising:

first program code configured to optimize execution of second program code in a computer of the type including a multi-level memory architecture by using a heuristic algorithm to select an ordering from among a plurality of orderings for a plurality of program code segments in the second program code; and

a signal bearing medium bearing the first program code.

31. The program product of claim 30, wherein the signal bearing medium includes at least one of a recordable medium and a transmission medium.